NoProp: Training Neural Networks without Back-propagation or Forward-propagation

Qinyu Li
Department of Statistics
University of Oxford
qinyu.li@stats.ox.ac.uk

Yee Whye Teh
Department of Statistics
University of Oxford
y.w.teh@stats.ox.ac.uk

Razvan Pascanu
Mila
r.pascanu@gmail.com

Abstract

The canonical deep learning approach for learning requires computing a gradient term at each layer by back-propagating the error signal from the output towards each learnable parameter. Given the stacked structure of neural networks, where each layer builds on the representation of the layer below, this approach leads to hierarchical representations. More abstract features live on the top layers of the model, while features on lower layers are expected to be less abstract. In contrast to this, we introduce a new learning method named *NoProp*, which does not rely on either forward or backwards propagation. Instead, *NoProp* takes inspiration from diffusion and flow matching methods, where each layer independently learns to denoise a noisy target. We believe this work takes a first step towards introducing a new family of gradient-free learning methods, that does not learn hierarchical representations – at least not in the usual sense. *NoProp* needs to fix the representation at each layer beforehand to a noised version of the target, learning a local denoising process that can then be exploited at inference. We demonstrate the effectiveness of our method on MNIST, CIFAR-10, and CIFAR-100 image classification benchmarks. Our results show that *NoProp* is a viable learning algorithm which achieves superior accuracy, is easier to use and computationally more efficient compared to other existing back-propagation-free methods. By departing from the traditional gradient based learning paradigm, *NoProp* alters how credit assignment is done within the network, enabling more efficient distributed learning as well as potentially impacting other characteristics of the learning process.

1 Introduction

Back-propagation (Rumelhart et al., 1986) has long been a cornerstone of deep learning, and its application has enabled deep learning technologies to achieve remarkable successes across a wide range of domains from sciences to industries. Briefly, it is an iterative algorithm, which adapts the parameters of a multi-layer neural network at each step such that its output match better a desired target. Each step of back-propagation first performs a forward-propagation of the input signal to generate a prediction, then compares the prediction to a desired target, and finally propagates the error signal back through the network to determine how the weights of each layer should be adjusted to decrease the error. This way, each layer can be thought of as learning to change the representation it receives from the layer below it to one that subsequent layers can use to make better predictions. The error signal propagated backwards is used to do *credit assignment*, i.e. to decide how much each parameter needs to change in order to minimize the error.

The simplicity of back-propagation has made it the de facto method for training neural networks. However over the years there has been consistent interest in developing alternative methods that do not rely on back-propagation. This interest is driven by several factors. Firstly, back-propagation is biologically implausible, as it requires synchronised alternation between forward and backward passes (e.g Lee et al., 2015). Secondly, back-propagation requires storing intermediate activations during the forward pass to facilitate gradient computation in the backward pass, which can impose significant memory overheads (Rumelhart et al., 1986). Finally, the sequential propagation of gradients introduces dependencies that impede parallel computation, making it challenging to effectively utilize multiple devices and servers for large-scale machine learning (Carreira-Perpinan & Wang, 2014). This sequential nature of how the credit assignment is computed has also additional implications for learning, leading to interference (Schaul et al., 2019) and playing a role in catastrophic forgetting (Hadsell et al., 2020).

Alternative optimization methods to back-propagation include gradient-free methods (e.g., direct search methods (Fermi, 1952; Torczon, 1991) and model-based methods (Bortz & Kelley, 1997; Conn et al., 2000)), zero-order gradient methods (Flaxman et al., 2004; Duchi et al., 2015; Nesterov & Spokoiny, 2015; Liu et al., 2020; Ren et al.,

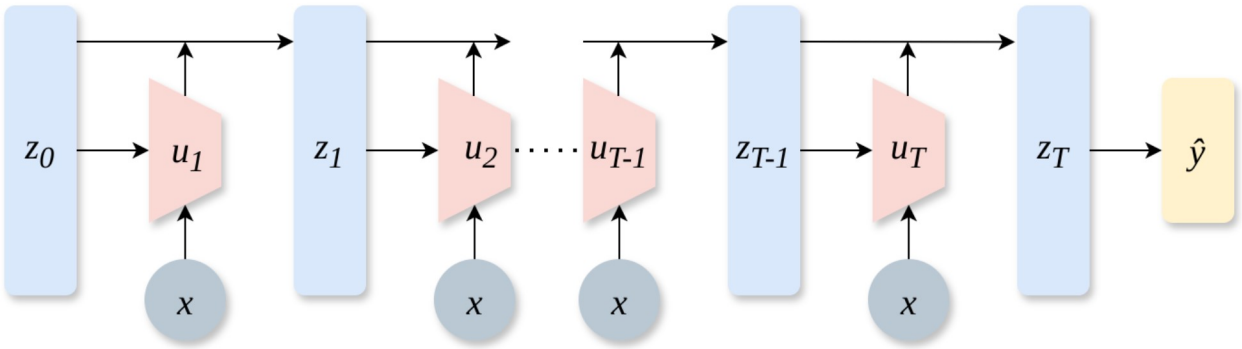

Figure 1: Architecture of NoProp. z_0 represents Gaussian noise, while $z_1, \ldots, z_T$ are successive transformations of z_0 through the learned dynamics $u_1, \ldots, u_T$, with each layer conditioned on the image x, ultimately producing the class prediction $\hat{y}$.

2022) that use finite difference to approximate gradients, evolution strategies (Wierstra et al., 2014; Salimans et al., 2017; Such et al., 2017; Khadka & Tumer, 2018), methods that rely on local losses such as difference target propagation (Lee et al., 2015) and the forward-forward algorithm (Hinton, 2022). However, these methods have not been widely adopted for learning neural networks due to their limitations in terms of accuracy, computational efficiency, reliability and scalability.

In this paper, we propose a back-propagation-free approach for training. The method is based on the denoising score matching approach that underlies diffusion models (Sohl-Dickstein et al., 2015; Chen et al., 2018; Ho et al., 2020; Song et al., 2021), enabling each layer of the neural network to be trained independently. In brief, at training time each layer is trained to predict the target label given a noisy label and the training input, while at inference time each layer takes the noisy label produced by the previous layer, and denoises it by taking a step towards the label it predicts. Of particular note is the observation that at training time the method does not even require a forward pass, hence we call our method *NoProp*. We show experimentally on the MNIST, CIFAR-10 and CIFAR-100 benchmarks that NoProp is significantly more performant than previous back-propagation-free methods while at the same time being simpler, more robust, and more computationally efficient.

2 Methodology

In this section we will describe the NoProp method for back-propagation-free learning. While the technical ideas follow closely those in variational diffusion models (Sohl-Dickstein et al., 2015; Kingma et al., 2021; Gulrajani & Hashimoto, 2024), we apply them in a different context and interpret them differently.

2.1 NoProp

Let x and y be a sample input-label pair in our classification dataset, assumed drawn from a data distribution $q_0(x, y)$, and $z_0, z_1, \ldots, z_T \in \mathbb{R}^d$ be corresponding *stochastic* intermediate activations of a neural network with T blocks, which we aim to train to estimate $q_0(y|x)$.

We define two distributions p and q decomposed in the following ways:

$$p((z_t)_{t=0}^T, y|x) = p(z_0) \left(\prod_{t=1}^T p(z_t|z_{t-1}, x) \right) p(y|z_T), \tag{1}$$

$$q((z_t)_{t=0}^T|y, x) = q(z_T|y) \left(\prod_{t=T}^1 q(z_{t-1}|z_t) \right). \tag{2}$$

The distribution p can be interpreted as a stochastic forward propagation process which iteratively computes the next activation z_t given the previous one z_{t-1} and the input x. In fact, we shall see that it can be explicitly given as a residual network with Gaussian noise added to the activations:

$$z_t = a_t \hat{u}_{\theta_t}(z_{t-1}, x) + b_t z_{t-1} + \sqrt{c_t}\epsilon_t, \quad \epsilon_t \sim \mathcal{N}_d(\epsilon_t|0, 1) \tag{3}$$

where $\mathcal{N}_d(\cdot|0, 1)$ is a d-dimensional Gaussian density with mean vector 0 and identity covariance matrix, a_t, b_t, c_t are scalars (given below), $b_t z_{t-1}$ is a weighted skip connection, and $\hat{u}_{\theta_t}(z_{t-1}, x)$ being a residual block parameterised by

θ_t. Note that this computation structure is different from a standard deep neural network which does not have direct connections from the input x into each block.

Following the variational diffusion model approach, we can alternatively interpret p as a conditional latent variable model for y given x, with the z_t's being a sequence of latent variables. We can learn the forward process p using a variational formulation, with the q distribution serving as the variational posterior. The objective of interest is then the ELBO, a lower bound of the log likelihood $\log p(y|x)$ (a.k.a. the evidence):

$$\log p(y|x) \geq \mathbb{E}_{q((z_t)_{t=0}^T|y,x)} \left[\log p((z_t)_{t=0}^T, y|x) - \log q((z_t)_{t=0}^T|y,x)\right]. \tag{4}$$

See Appendix A.1 for the derivation of Equation 4. Following Sohl-Dickstein et al. (2015); Kingma et al. (2021), we fix the variational posterior q to a tractable Gaussian distribution; here we use the variance preserving Ornstein-Uhlenbeck process:

$$q(z_T|y) = \mathcal{N}_d(z_T|\sqrt{\alpha_T}u_y, 1-\alpha_T), \qquad q(z_{t-1}|z_t) = \mathcal{N}_d(z_{t-1}|\sqrt{\alpha_{t-1}}z_t, 1-\alpha_{t-1}) \tag{5}$$

where u_y is an embedding of the class label y in $\mathbb{R}^d$, defined by a trainable embedding matrix $W_{\text{Embed}} \in \mathbb{R}^{m\times d}$, with m as the number of classes. The embedding is given by $u_y = \{W_{\text{Embed}}\}_y$. Using standard properties of the Gaussian, we can obtain

$$q(z_t|y) = \mathcal{N}_d(z_t|\sqrt{\bar{\alpha}_t}u_y, 1-\bar{\alpha}_t), \qquad q(z_t|z_{t-1}, y) = \mathcal{N}_d(z_t|\mu_t(z_{t-1}, u_y), c_t) \tag{6}$$

with $\bar{\alpha}_t = \prod_{s=t}^T \alpha_s$, $\mu_t(z_{t-1}, u_y) = a_t u_y + b_t z_{t-1}$, $a_t = \frac{\sqrt{\bar{\alpha}_t}(1-\alpha_{t-1})}{1-\bar{\alpha}_{t-1}}$, $b_t = \frac{\sqrt{\alpha_{t-1}}(1-\bar{\alpha}_t)}{1-\bar{\alpha}_{t-1}}$, and $c_t = \frac{(1-\bar{\alpha}_t)(1-\alpha_{t-1})}{1-\bar{\alpha}_{t-1}}$. See Appendix A.2 and A.3 for a full derivation. To optimise the ELBO, we parameterise p to match the form for q:

$$p(z_t|z_{t-1}, x) = \mathcal{N}_d(z_t|\mu_t(z_{t-1}, \hat{u}_{\theta_t}(z_{t-1}, x)), c_t), \qquad p(z_0) = \mathcal{N}_d(z_0|0, 1) \tag{7}$$

where $p(z_0)$ has been chosen to be the stationary distribution of the Ornstein-Uhlenbeck process, and $\hat{u}_{\theta_t}(z_{t-1}, x)$ is a neural network block parameterised by θ_t. The resulting computation to sample z_t given z_{t-1} and x is as given in by the residual architecture (Equation 3), with a_t, b_t, c_t as specified above. Finally, plugging this parameterisation into the ELBO (Equation 4) and simplifying, we obtain the NoProp objective,

$$\begin{aligned}\mathcal{L}_{\text{NoProp}} =& \mathbb{E}_{q(z_T|y)}\left[-\log \hat{p}_{\theta_{\text{out}}}(y|z_T)\right] \\ &+ D_{\text{KL}}(q(z_0|y)\|p(z_0)) \\ &+ \frac{T}{2}\eta \mathbb{E}_{t\sim\mathcal{U}\{1,T\}}\left[(\text{SNR}(t) - \text{SNR}(t-1))\|\hat{u}_{\theta_t}(z_{t-1}, x) - u_y\|^2\right],\end{aligned} \tag{8}$$

where $\text{SNR}(t) = \frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}$ is the *signal-to-noise ratio*, η is a hyperparameter, and $\mathcal{U}\{1, T\}$ is the uniform distribution on the integers $1, \ldots, T$. See Appendix A.4 for a full derivation of the NoProp objective.

We see that each $\hat{u}_{\theta_t}(z_{t-1}, x)$ is trained to directly predict u_y given z_{t-1} and x with an L2 loss, while $\hat{p}_{\theta_{\text{out}}}(y|z_T)$ is trained to minimise cross-entropy loss. Each block $\hat{u}_{\theta_t}(z_{t-1}, x)$ is trained independently, and this is achieved without forward- or back-propagation across the network.

2.2 Variations

2.2.1 Fixed and learnable W_{Embed}

The class embedding matrix W_{Embed} can be either fixed or jointly learned with the model. In the fixed case, we set W_{Embed} to be the identity matrix, where each class embedding is a one-hot vector. In the learned case, W_{Embed} is initialized as an orthogonal matrix when possible, or randomly otherwise. When the embedding dimension matches the image dimension, we interpret this as learning an "image prototype." In this special case, class embeddings are initialized as the images with the smallest median distance to all other images within the same class, serving as crude prototypes before training.

2.2.2 Continuous-time diffusion models and neural ODEs

The NoProp framework presented above involves learning $\hat{u}_{\theta_t}(z_{t-1}, x)$ parameterised by θ_t for each layer $t = 1, 2, \ldots, T$ along with a linear layer for $\hat{p}_{\theta_{\text{out}}}(y|z_T)$. We can extend this to a continuous version where the number of latent variables T tends to infinity (Kingma et al., 2021), leading to the objective

$$\mathbb{E}_{q(z_1|y)}\left[-\log \hat{p}_{\theta_{\text{out}}}(y|z_1)\right] + D_{\text{KL}}(q(z_0|y)\|p(z_0)) + \frac{1}{2}\eta \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\text{SNR}'(t)\|\hat{u}_\theta(z_t, x, t) - u_y\|^2\right]. \tag{9}$$

See Appendix A.4 for a full derivation. Here, time t is treated as a continuous variable in $[0, 1]$. We use a continuous noise schedule and $\text{SNR}'(t)$ denotes the derivative of the signal-to-noise ratio. A single block $\hat{u}_\theta(z_t, x, t)$ is trained for all time steps t, with t as an additional input. The evolution of latent variables z_t follows a continuous-time diffusion process, described by a stochastic differential equation (SDE). Note that there exits a corresponding ordinary differential equation (ODE) $\frac{d}{dt}z_t = f(z_t|x, t)$ sharing the same marginal distributions as the SDE (Song et al., 2021). The function $f(z_t|x, t)$ represents the deterministic vector field generating z_t's evolution. Neural networks parameterised in this manner are known as neural ODEs (Chen et al., 2018).

Neural ODE training usually relies on back-propagating through the ODE solver to optimize task-specific loss functions (e.g., cross-entropy in classification), or using the adjoint sensitivity method (Chen et al., 2018) which estimates the gradient by solving another ODE backward in time. The continuous-time diffusion model instead learns an ODE dynamic that inverts a predefined noising process. Training occurs by sampling time steps independently, without requiring full forward or backward passes through time. This makes training more efficient while still enabling expressive ODE dynamics.

2.2.3 Flow matching

An alternative approach to NoProp's continuous formulation is flow matching (Lipman et al., 2022; Tong et al., 2023), which directly learns the vector field $f(z_t|x, t)$ that transports noise toward the predicted label embedding via an ODE. Since $f(z_t|x, t)$ is generally unknown, flow matching instead learns the conditional vector field $f(z_t|z_0, z_1, x, t)$, where z_0 is the initial noise and $z_1 = u_y$ is the label embedding. The conditional vector field is user-specified. A simple choice is to define a Gaussian probability path between noise z_0 and label embedding $z_1 = u_y$

$$p_t(z_t|z_0, z_1, x) = \mathcal{N}_d(z_t \mid tz_1 + (1 - t)z_0, \sigma^2), \tag{10}$$

with the corresponding vector field $f(z_t|z_0, z_1, x, t) = z_1 - z_0$. The flow matching objective is then

$$\mathbb{E}_{t\sim\mathcal{U}[0,1],q_0(x,y),p(z_0),p_t(z_t|z_0,z_1,x)}\|v_\theta(z_t, x, t) - (z_1 - z_0)\|^2, \tag{11}$$

where $v_\theta(z_t, x, t)$ is a neural network with parameters θ. When the label embeddings are jointly learned, we introduce an additional anchor loss to prevent different class embeddings from collapsing. Following prior work (Gao et al., 2022; Hu et al., 2024), we incorporate a cross-entropy term $-\log \hat{p}_{\theta_{\text{out}}}(y \mid \tilde{z}_1(z_t, x, t))$, where $\tilde{z}_1(z_t, x, t)$ is an extrapolated linear estimate

$$\tilde{z}_1(z_t, x, t) = z_t + (1 - t)v_\theta(z_t, x, t). \tag{12}$$

The modified flow matching objective is

$$\mathbb{E}_{t\sim\mathcal{U}[0,1],q_0(x,y),p(z_0),p_t(z_t|z_0,z_1,x)}[\|v_\theta(z_t, x, t) - (z_1 - z_0)\|^2 - \log \hat{p}_{\theta_{\text{out}}}(y \mid \tilde{z}_1(z_t, x, t))]. \tag{13}$$

Similarly to diffusion, we parameterise $\hat{p}_{\theta_{\text{out}}}(y \mid \tilde{z}_1(z_t, x, t))$ using a linear layer followed by softmax. This formulation ensures well-separated class embeddings.

2.3 Implementation details

2.3.1 Architecture

The NoProp architecture is illustrated in Figure 1. During inference, Gaussian noise z_0 undergoes a sequence of transformations through the diffusion process. At each step, the latent variable z_t evolves via a diffusion dynamic block u_t, producing a sequence $z_1, z_2, \ldots, z_T$ until reaching z_T. Each u_t is conditioned on the previous latent state z_{t-1} and the input image x. Finally, a linear layer followed by a softmax function maps z_T to the predicted label $\hat{y}$. The model used to parameterise each u_t block is described in Section 2.3.2.

This architecture is designed specifically for inference. During training, time steps are sampled, and each diffusion dynamic block u_t is trained independently. The linear layer and the embedding matrix are trained jointly with the diffusion blocks, as the linear layer helps prevent the class embeddings from collapsing.

For the flow matching variant, the u_t's in Figure 1 represent the ODE dynamics. The label prediction $\hat{y}$ is obtained directly from z_T by finding the class embedding closest to z_T in terms of Euclidean distance.

2.3.2 Training procedure

Models The models used for training are illustrated in Figure 6 in the Appendix. For discrete-time diffusion, we use a neural network $\hat{u}_{\theta_t}$ to model the diffusion dynamics u_t. The model takes an input image x and a latent variable

z_{t-1} in the label embedding space, processing them through separate embedding pathways before concatenation. The image x is passed through a convolutional embedding module followed by a fully connected layer. When the embedding dimension matches the image dimension, z_{t-1} is treated as an image and embedded similarly using a convolutional module. Otherwise, z_{t-1} is passed through a fully connected network with skip connections. The fused representation is then processed by additional fully connected layers, producing logits as output. The diffusion dynamics $\hat{u}_{\theta_t}$ are obtained by applying a softmax function to the logits, yielding a probability distribution over class embeddings. The final output $\hat{u}_{\theta_t}$ is then computed as a weighted sum of the class embeddings in W_{Embed} using this probability distribution.

For continuous-time diffusion, we train $\hat{u}_\theta$, which takes an additional timestamp t as input. The timestamp is encoded using positional embeddings and processed through a fully connected layer before being concatenated with those of x and z_t. The rest of the model follows roughly the same structure as in the discrete case.

For flow matching, we train a neural network $\hat{v}_\theta$. The architecture remains the same as in the continuous-time diffusion case, but instead of applying softmax, $\hat{v}_\theta$ is obtained by directly computing the weighted sum of the class embeddings using the logits, treating the logits as unrestricted weights. This allows $\hat{v}_\theta$ to represent arbitrary directions in the embedding space, unlike $\hat{u}_\theta$, which is constrained to a convex combination of class embeddings.

While the overall model structure remains similar across settings, the inclusion of t in the continuous-time diffusion and flow matching cases introduces an additional input and processing step compared to the discrete-time diffusion case. We choose to parameterise $\hat{p}_{\theta_{\text{out}}}(y|z_T)$ in Equation 8, $\hat{p}_{\theta_{\text{out}}}(y|z_1)$ in Equation 9, and $\hat{p}_{\theta_{\text{out}}}(y \mid \tilde{z}_1(z_t, x, t))$ in Equation 13 using a linear layer followed by softmax.

Noise schedule For discrete-time diffusion, we use a fixed cosine noise schedule. For continuous-time diffusion, we train the noise schedule jointly with the model. Further details on the trainable noise schedule are provided in Appendix B.

3 Related Works

3.1 Back-propagation-Free Methods

Forward-forward algorithm (Hinton, 2022) replaces the traditional forward and backward passes with two forward passes in image classification: one using images paired with correct labels and the other with images paired with incorrect labels.

Target propagation and its variant (e.g. Lee et al., 2015; Lillicrap et al., 2016) work by learning an inverse mapping from the output layer to the hidden layers, producing local targets for weight updates without back-propagation. However, their performance heavily depends on the quality of the learned inverse mappings, and inaccuracies in the autoencoders can lead to suboptimal updates, limiting their effectiveness in deep networks.

Zero-order methods (Flaxman et al., 2004; Duchi et al., 2015; Ren et al., 2022) is a broad family of algorithms that estimate gradients without explicit differentiation, making them useful for black-box and non-differentiable functions.

Evolution strategies (Wierstra et al., 2014; Salimans et al., 2017; Khadka & Tumer, 2018), inspired by natural evolution, optimize parameters by perturbing them with random noise and updating in the direction of higher rewards using a population-based optimization approach. However, they require a large number of function evaluations, leading to high sample complexity, and effective exploration remains a key challenge for high-dimensional parameter spaces.

3.2 Diffusion and flow matching

Several works in diffusion and flow matching are closely related to our method. Han et al. (2022) introduced classification and regression diffusion models, Kim et al. (2025) proposed flow matching for paired data, and Hu et al. (2024) applied flow matching to conditional text generation. In contrast, our paper explores the implications of these ideas within the framework of back-propagation-free learning.

3.3 Representation Learning

Generally speaking, most alternative methods to back-propagation, whether they simply approximate gradients differently, or utilize a different search scheme — as it is the case of evolution strategies — still rely on learning intermediate representations that build on top of each other (Bengio et al., 2013). This allows learning more abstract representations as we look at deeper layers of the model, and is thought to be of fundamental importance for deep learning and for

representing cognitive processes (Markman & Dietrich, 2000). Indeed, initial successes of deep learning have been attributed to the ability to train *deep* architectures to learn hierarchical representations (Hinton et al., 2006; Bengio et al., 2013), while early interpretability work focused on visualising these increasingly more complex features (Zeiler & Fergus, 2014; Lee et al., 2009).

By getting each layer to learn to denoise labels, with the label noise distribution chosen by the user, we can argue that NoProp does not *learn* representations. Rather, it relies on representations *designed* by the user (specifically, the representations in intermediate layers are Gaussian noised embeddings of target labels for diffusion, and an interpolation between Gaussian noise and embeddings of target labels for flow matching). This is perhaps unsurprising: forward- and back-propagation can be understood as information being disseminated across the layers of a neural network in order to enable the representation of each layer to "fit in" with those of neighbouring layers and such that the target label can be easily predicted from the representation at the last layer. So for NoProp to work without forward- or back-propagation, these layer representations have to be fixed beforehand, i.e. to be designed by the user.

The fact that NoProp achieves good performance without representation learning leads to the question of whether representation learning is in fact necessary for deep learning, and whether by designing representations we can enable alternative approaches to deep learning with different characteristics. Specifically, note that representations fixed in NoProp are not those one might think of as being more abstract in later layers, which opens the door to revisiting the role of hierarchical representations in modelling complex behaviour (Markman & Dietrich, 2000). These questions can become increasingly important as core assumptions of back-propagation based learning, like the i.i.d. assumption and sequential propagation of information and error signal through the network, are proving to be limiting.

4 Experiments

We compare NoProp against back-propagation in the discrete-time case and against the adjoint sensitivity method (Chen et al., 2018) in the continuous-time case for image classification tasks. Details on the hyperparameters are provided in Table 3 in the Appendix.

Datasets. We use three benchmark datasets: MNIST, CIFAR-10, and CIFAR-100. The MNIST dataset consists of 70,000 grayscale images of handwritten digits (28x28 pixels), spanning 10 classes (digits 0-9). CIFAR-10 contains 60,000 color images (32x32 pixels) across 10 different object classes. CIFAR-100, similar in structure to CIFAR-10, includes 60,000 color images, but it is divided into 100 fine-grained classes. We use the standard training/test set splits. In our experiments, we do not apply any data augmentation techniques to these datasets.

NoProp (discrete-time) We refer to this method as NoProp with Discrete-Time Diffusion (NoProp-DT). We fix $T = 10$ and, during each epoch, update the parameters for each time step sequentially, as outlined in Algorithm 1. While sequential updates are not strictly necessary for the algorithm to work — as time steps can also be sampled — we choose this approach to align with other back-propagation-free methods, ensuring consistency with existing approaches in the literature. In terms of parameterisation of the class probabilities $\hat{p}_{\theta_{\text{out}}}(y|z_T)$ in Equation 8, we explore two approaches. The first, as described earlier, involves using softmax on the outputs of a fully connected layer f with parameters θ_{out}, such that

$$\hat{p}_{\theta_{\text{out}}}(y \mid z_T) = \text{softmax}(f_{\theta_{\text{out}}}(z_T)) = \frac{\exp(f_{\theta_{\text{out}}}(z_T)_y)}{\sum_{y'=1}^{m} \exp(f_{\theta_{\text{out}}}(z_T)_{y'})}, \tag{14}$$

where m is the number of classes. As an additional exploration, we investigate an alternative approach that relies on radial distance to parameterise the class probabilities. In this approach, we estimate a reconstructed label $\tilde{y}$ from z_T by applying softmax to the output of the same fully connected layer $f_{\theta_{\text{out}}}$, followed by a projection onto the class embedding matrix W_{Embed}, such that $\tilde{y}$ is a weighted embedding. The resulting probability of class y given z_T is then based on the squared Euclidean distance between $\tilde{y}$ and the true class embedding $u_y = (W_{\text{Embed}})_y$:

$$\tilde{y} = \text{softmax}(f_{\theta_{\text{out}}}(z_T))W_{\text{Embed}}, \tag{15}$$

$$\hat{p}_{\theta_{\text{out}}}(y|z_T) = \frac{\exp\left(-\frac{\|\tilde{y}-u_y\|^2}{2\sigma^2}\right)}{\sum_{y'=1}^{m} \exp\left(-\frac{\|\tilde{y}-u_{y'}\|^2}{2\sigma^2}\right)}. \tag{16}$$

This formulation can be interpreted as computing the posterior class probability assuming equal prior class probabilities and a normal likelihood $p(\tilde{y}|y) = \mathcal{N}_d(\tilde{y}|u_y, \sigma^2)$.

Method	MNIST		CIFAR-10		CIFAR-100	
	Train	Test	Train	Test	Train	Test
Discrete-time						
Backprop (one-hot)	100.0±0.0	99.46±0.06	99.98±0.01	79.92±0.14	98.63±1.34	45.85±2.07
Backprop (dim=20)	99.99±0.0	99.43±0.03	99.96±0.02	79.3±0.52	94.28±7.43	46.57±0.87
Backprop (prototype)	99.99±0.01	99.44±0.05	99.97±0.01	79.58±0.44	99.19±0.71	**47.8±0.19**
NoProp-DT (one-hot)	99.92±0.01	99.47±0.05	95.02±0.19	79.25±0.28	84.97±0.67	45.93±0.46
NoProp-DT (dim=20)	99.93±0.01	99.49±0.04	94.95±0.09	79.12±0.37	83.25±0.39	45.19±0.22
NoProp-DT (prototype)	99.97±0.0	**99.54±0.04**	97.23±0.11	**80.54±0.2**	90.7±0.14	46.06±0.25
Continuous-time						
Adjoint (one-hot)	98.7±0.13	98.62±0.14	70.64±0.49	66.78±0.76	26.72±0.81	25.03±0.7
NoProp-CT (one-hot)	97.88±0.61	97.84±0.71	97.31±0.84	73.35±0.55	75.1±3.43	33.66±0.5
NoProp-CT (dim=20)	97.7±0.42	97.7±0.51	94.88±3.08	71.77±2.47	74.22±2.33	33.99±1.08
NoProp-CT (prototype)	97.18±1.02	97.17±0.94	86.2±7.34	66.54±3.63	40.88±10.72	21.31±4.17
NoProp-FM (one-hot)	99.97±0.0	99.21±0.09	98.46±0.4	73.14±0.9	12.69±10.4	6.38±4.9
NoProp-FM (dim=20)	99.99±0.0	**99.29±0.05**	99.49±0.15	73.5±0.28	83.49±4.62	31.14±0.52
NoProp-FM (prototype)	99.27±0.09	98.52±0.16	99.8±0.03	**75.18±0.57**	96.37±1.09	**37.57±0.32**
Previous back-propagation-free methods						
Forward-Forward	-	98.63	-	-	-	-
Forward Gradient	100.00	97.45	80.61	69.32	-	-
Difference Target Prop	-	98.06	-	50.71	-	-

Table 1: Classification accuracies (%) on MNIST, CIFAR-10, and CIFAR-100. Means and standard errors are computed from 15 values per method, obtained from 3 seeds and 5 inference runs per seed. Forward Gradient refers to the Local Greedy Forward Gradient Activity-Perturbed (LG-FG-A) method in Ren et al. (2022). one-hot: fixed one-hot label embedding; dim=20: learned label embedding of dimension 20; prototype: learned label embedding of dimension equal to image size.

Back-propagation. We also fix $T = 10$ and the forward pass is given by

$$z_0 \sim \mathcal{N}_d(z_0|0, 1), \tag{17}$$

$$z_t = (1 - \alpha_t)z_{t-1} + \alpha_t \hat{u}_{\theta_t}(z_{t-1}, x), \quad t = 1, \ldots, T, \tag{18}$$

$$\hat{y} = \text{argmax}_{i \in \{1,\ldots,m\}} \hat{p}_{\theta_{\text{out}}}(y_i|z_T), \tag{19}$$

where $\alpha_1, \ldots, \alpha_T$ are learnable parameters constrained to the range $(-1, 1)$, defined as $\alpha_t = \tanh(w_t)$ with learnable $w_1, \ldots, w_T$. This design closely resembles the forward pass of NoProp-DT, but with the network trained by standard back-propagation. The $\hat{u}_{\theta_t}(z_{t-1}, x)$ and $\hat{p}_{\theta_{\text{out}}}(y|z_T)$ have identical model structures to those in NoProp-DT.

NoProp (continuous-time) We refer to the continuous-time variants as NoProp with Continuous-Time Diffusion (NoProp-CT) and NoProp with Flow Matching (NoProp-FM). During training, the time variable t is randomly sampled from $[0, 1]$. During inference, we set $T = 1000$ steps to simulate the diffusion process for NoProp-CT and the probability flow ODE for NoProp-FM. Detailed training procedures are provided in Algorithms 2 and 3.

Adjoint sensitivity We also evaluate the adjoint sensitivity method (Chen et al., 2018), which trains a neural ODE using gradients estimated by solving a related adjoint equation backward in time. Including this method provides a meaningful baseline to assess the efficiency and accuracy of our approach in the continuous-time case. For fair comparison, we fix $T = 1000$ during both training and inference and use a linear layer for final classification.

Main results Our results, summarised in Table 1, demonstrate that NoProp-DT achieves performance comparable to or better than back-propagation on MNIST, CIFAR-10, and CIFAR-100 in the discrete-time setting. Moreover, NoProp-DT outperforms prior back-propagation-free methods, including the Forward-Forward Algorithm, Difference Target Propagation (Lee et al., 2015), and a forward gradient method referred to as Local Greedy Forward Gradient Activity-Perturbed (Ren et al., 2022). While these methods use different architectures and do not condition layers explicitly on image inputs as NoProp does—making direct comparisons challenging—our method offers the distinct advantage of not requiring a forward pass. Additionally, NoProp demonstrates reduced GPU memory consumption

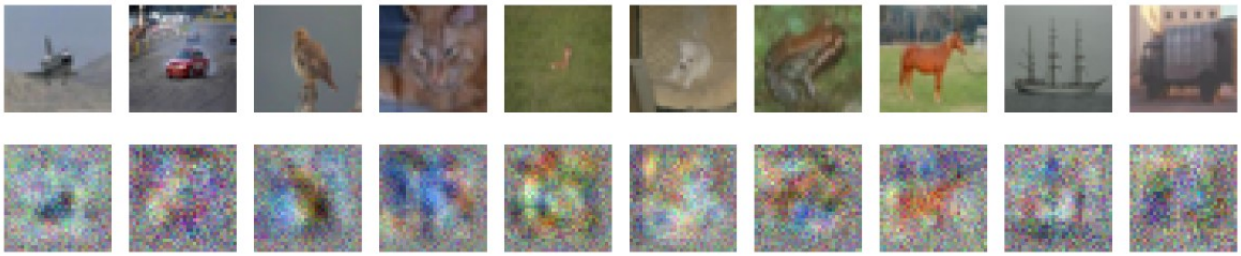

Figure 2: The first row shows class embeddings initialized using the image with the smallest median distance to all other images within the same class for CIFAR-10. The second row displays the learned class embeddings from NoProp-DT, which can be interpreted as learned image prototypes for each class.

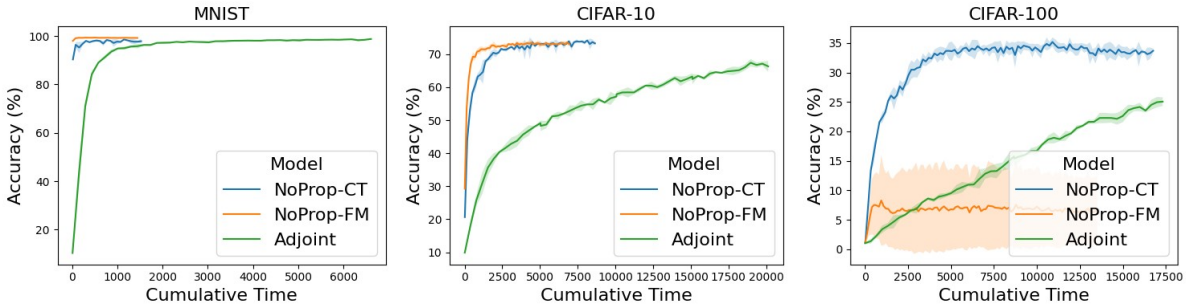

Figure 3: Test accuracies (%) plotted against cumulative training time (in seconds) for models using one-hot label embedding in the continuous-time setting. All models within each plot were trained on the same type of GPU to ensure a fair comparison. NoProp-CT achieves strong performance in terms of both accuracy and speed compared to adjoint sensitivity. For CIFAR-100, NoProp-FM does not learn effectively with one-hot label embedding.

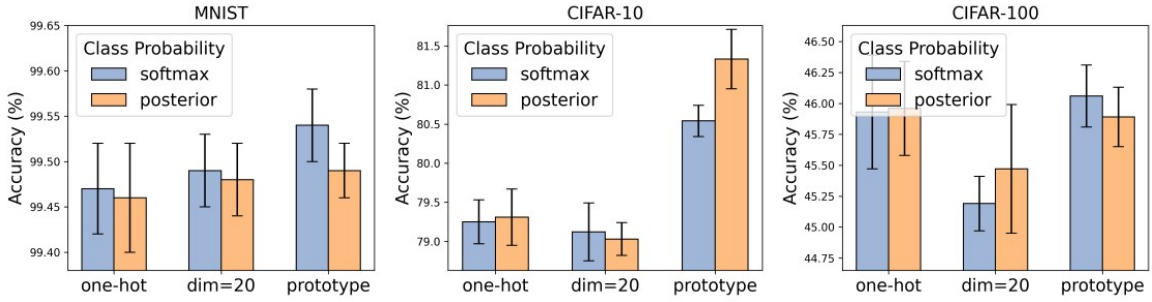

Figure 4: Test accuracy (%) comparison for two parameterisations of class probabilities $\hat{p}_{\theta_{\text{out}}}(y|z_T)$ in Equation 8, using either softmax or posterior probability.

Method	MNIST	CIFAR-10	CIFAR-100
Discrete-time			
Backprop	0.87 GB	1.17 GB	1.73 GB
NoProp-DT	0.49 GB	0.64 GB	1.23 GB
Continuous-time			
Adjoint	2.32 GB	6.23 GB	6.45 GB
NoProp-CT	1.05 GB	0.45 GB	0.50 GB
NoProp-FM	1.06 GB	0.44 GB	0.49 GB

Table 2: Process GPU memory allocated (in GB) for models using one-hot label embedding.

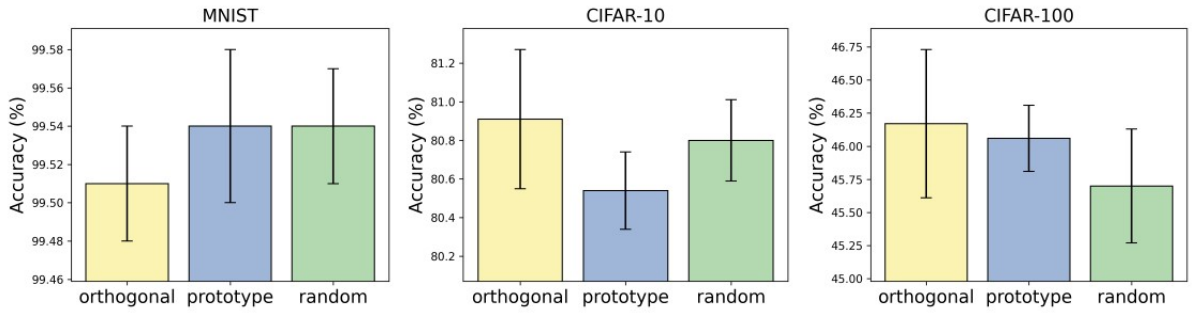

Figure 5: Test accuracy (%) comparison for different initializations of the class embedding matrix W_{Embed} when the class embedding dimension matches the image dimension. The initializations considered are random matrices, orthogonal matrices, and prototype images.

during training, as shown in Table 2. To illustrate the learned class embeddings, Figure 2 visualises both the initializations and the final class embeddings for CIFAR-10 learned by NoProp-DT, where the embedding dimension matches the image dimension.

In the continuous setting, NoProp-CT and NoProp-FM achieve lower accuracy than NoProp-DT, likely due to the additional conditioning on time t. However, they generally outperform the adjoint sensitivity method on CIFAR-10 and CIFAR-100, both in terms of accuracy and computational efficiency. While the adjoint method achieves similar accuracy to NoProp-CT and NoProp-FM on MNIST, it does so much slower, as shown in Figure 3.

For CIFAR-100 with one-hot embeddings, NoProp-FM fails to learn effectively, resulting in very slow accuracy improvement. In contrast, NoProp-CT still outperforms the adjoint method. However, once label embeddings are learned jointly, the performance of NoProp-FM improves significantly.

We also conducted ablation studies on the parameterisations of class probabilities, $\hat{p}_{\theta_{\text{out}}}(y|z_T)$, and the initializations of the class embedding matrix, W_{Embed}, with results shown in Figure 4 and Figure 5, respectively. The ablation results reveal no consistent advantage between the class probability parameterisations, with performance varying across datasets. For class embedding initializations, both orthogonal and prototype initializations are generally comparable to or outperform random initializations.

5 Conclusion

Using the denoising score matching approach that underlies diffusion models, we have proposed NoProp, a forward- and back-propagation-free approach for training neural networks. The method enables each layer of the neural network to be trained independently to predict the target label given a noisy label and the training input, while at inference time, each layer takes the noisy label produced by the previous layer and denoises it by taking a step towards the label it predicts. We show experimentally that NoProp is significantly more performant than previous back-propagation-free methods while at the same time being simpler, more robust, and more computationally efficient. We believe that this perspective of training neural networks via denoising score matching opens up new possibilities for training deep learning models without back-propagation, and we hope that our work will inspire further research in this direction.

REFERENCES

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013. ISSN 0162-8828.

David Bortz and Carl Kelley. The simplex gradient and noisy optimization problems. *Computational Methods for Optimal Design and Control*, 02 1997.

Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In Samuel Kaski and Jukka Corander (eds.), *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pp. 10–19, Reykjavik, Iceland, 22–25 Apr 2014. PMLR.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Andrew Conn, Nicholas Gould, and Philippe Toint. Trust region methods. *SIAM*, 2000.

John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

E. Fermi. Numerical solution of a minimum problem. Technical report, Los Alamos Scientific Lab., Los Alamos, NM, 11 1952.

Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. *arXiv preprint cs/0408007*, 2004.

Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. Difformer: Empowering diffusion models on the embedding space for text generation. *arXiv preprint arXiv:2212.09412*, 2022.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, 2020.

Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. Card: Classification and regression diffusion models. *Advances in Neural Information Processing Systems*, 35:18100–18115, 2022.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Vincent Hu, Di Wu, Yuki Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees Snoek. Flow matching for conditional text generation in a few sampling steps. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 380–392, 2024.

Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Semin Kim, Jaehoon Yoo, Jinwoo Kim, Yeonwoo Cha, Saehoon Kim, and Seunghoon Hong. Simulation-free training of neural odes on paired data. *Advances in Neural Information Processing Systems*, 37:60212–60236, 2025.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pp. 498–515. Springer, 2015.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 609–616, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553453. URL https://doi.org/10.1145/1553374.1553453.

Timothy P. Lillicrap, Daniel Cownden, Douglas Blair Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, 2016. URL https://api.semanticscholar.org/CorpusID:10050777.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O. Hero III, and Pramod K. Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020.

Arthur B. Markman and Eric Dietrich. In defense of representation. *Cognitive Psychology*, 40(2):138–171, 2000.

Yurii Nesterov and Vladimir G. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527 – 566, 2015.

Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. *arXiv preprint arXiv:2210.03310*, 2022.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning, 2019.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021.

Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *ArXiv*, abs/1712.06567, 2017.

Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Virginia Torczon. On the convergence of the multidirectional search algorithm. *SIAM Journal on Optimization*, 1(1): 123–145, 1991.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 818–833, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10590-1.

A Derivations of training objectives of NoProp

For completeness, we include derivations of the training objectives of NoProp-DT and NoProp-CT, closely following Sohl-Dickstein et al. (2015) and Kingma et al. (2021).

A.1 Derivation of Equation 4

$$\log p(y|x) = \log \int p((z_t)_{t=0}^T, y|x) d(z_t)_{t=0}^T \tag{20}$$

$$= \log \int \frac{p((z_t)_{t=0}^T, y|x) q((z_t)_{t=0}^T|y,x)}{q((z_t)_{t=0}^T|y,x)} d(z_t)_{t=0}^T \tag{21}$$

$$= \log \mathbb{E}_{q((z_t)_{t=0}^T|y,x)} \left[\frac{p((z_t)_{t=0}^T, y|x)}{q((z_t)_{t=0}^T|y,x)} \right] \tag{22}$$

$$\geq \mathbb{E}_{q((z_t)_{t=0}^T|y,x)} \left[\log p((z_t)_{t=0}^T, y|x) - \log q((z_t)_{t=0}^T|y,x) \right]. \tag{23}$$

The last step follows from Jensen's inequality, yielding a lower bound on $\log p(y|x)$. This bound is commonly referred to as the evidence lower bound (ELBO).

A.2 $q(z_{t-1}|z_t)$ and $q(z_t|y)$

We will use the reparameterization trick to derive the expressions for $q(z_t|y)$ and $q(z_t|z_s)$. The reparameterization trick allows us to rewrite a random variable sampled from a distribution in terms of a deterministic function of a noise variable. Specifically, for a Gaussian random variable $z \sim \mathcal{N}_d(z|\mu, \sigma^2)$, we can reparameterize it as:

$$z = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}_d(\epsilon|0,1). \tag{24}$$

Let $\{\epsilon_t^*, \epsilon_t\}_{t=0}^T, \epsilon_y^*, \epsilon_y \sim \mathcal{N}_d(0,1)$. Then, for any sample $z_t \sim q(z_t|z_s)$ for any $0 \leq t < s \leq T$, it can be expressed as

$$z_t = \sqrt{\alpha_t} z_{t+1} + \sqrt{1-\alpha_t}\epsilon_{t+1}^* \tag{25}$$

$$= \sqrt{\alpha_t}(\sqrt{\alpha_{t+1}} z_{t+2} + \sqrt{1-\alpha_{t+1}}\epsilon_{t+2}^*) + \sqrt{1-\alpha_t}\epsilon_{t+1}^* \tag{26}$$

$$= \sqrt{\alpha_t\alpha_{t+1}} z_{t+2} + \sqrt{\alpha_t - \alpha_t\alpha_{t+1}}\epsilon_{t+2}^* + \sqrt{1-\alpha_t}\epsilon_{t+1}^* \tag{27}$$

$$= \sqrt{\alpha_t\alpha_{t+1}} z_{t+2} + \sqrt{(\alpha_t - \alpha_t\alpha_{t+1}) + (1-\alpha_t)}\epsilon_{t+2} \tag{28}$$

$$= \sqrt{\alpha_t\alpha_{t+1}} z_{t+2} + \sqrt{1-\alpha_t\alpha_{t+1}}\epsilon_{t+2} \tag{29}$$

$$= \sqrt{\prod_{i=t}^{s-1}\alpha_i} z_s + \sqrt{1-\prod_{i=t}^{s-1}\alpha_i}\epsilon_s \tag{30}$$

$$= \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_s}} z_s + \sqrt{1-\frac{\bar{\alpha}_t}{\bar{\alpha}_s}}\epsilon_s. \tag{31}$$

Hence,

$$q(z_t|z_s) = \mathcal{N}_d(z_t|\sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_s}}, 1-\frac{\bar{\alpha}_t}{\bar{\alpha}_s}). \tag{32}$$

In particular,

$$q(z_{t-1}|z_t) = \mathcal{N}_d(z_{t-1}|\sqrt{\alpha_{t-1}} z_t, 1-\alpha_{t-1}). \tag{33}$$

Similarly, it can be shown that

$$q(z_t|y) = \mathcal{N}_d(z_t|\sqrt{\bar{\alpha}_t} u_y, 1-\bar{\alpha}_t). \tag{34}$$

A.3 $q(z_t|z_{t-1}, y)$

Applying the Bayes' theorem, we can obtain the posterior density

$$
\begin{aligned}
q(z_t|z_{t-1}, y) &\propto q(z_t|y)q(z_{t-1}|z_t) && (35)\\
&\propto \mathcal{N}_d(z_t|\sqrt{\bar{\alpha}_t}u_y, 1-\bar{\alpha}_t)\mathcal{N}_d(z_{t-1}|\sqrt{\bar{\alpha}_{t-1}}z_t, 1-\alpha_{t-1}) && (36)\\
&\propto \exp\left(-\frac{1}{2(1-\bar{\alpha}_t)}(z_t-\sqrt{\bar{\alpha}_t}u_y)^T(z_t-\sqrt{\bar{\alpha}_t}u_y)\right. && (37)\\
&\left.-\frac{1}{2(1-\alpha_{t-1})}(z_{t-1}-\sqrt{\alpha_{t-1}}z_t)^T(z_{t-1}-\sqrt{\alpha_{t-1}}z_t)\right) && (38)\\
&\propto \exp\left(-\frac{1}{2c_t}(z_t-\mu_t(z_{t-1},u_y))^T(z_t-\mu_t(z_{t-1},u_y))\right). && (39)
\end{aligned}
$$

Hence we have $q(z_t|z_{t-1}, y) = \mathcal{N}_d(z_t|\mu_t(z_{t-1}, u_y), c_t)$, where

$$
\begin{aligned}
\mu_t(z_{t-1}, u_y) &= \frac{\sqrt{\bar{\alpha}_t}(1-\alpha_{t-1})}{1-\bar{\alpha}_{t-1}}u_y + \frac{\sqrt{\alpha_{t-1}}(1-\bar{\alpha}_t)}{1-\bar{\alpha}_{t-1}}z_{t-1}, && (40)\\
c_t &= \frac{(1-\bar{\alpha}_t)(1-\alpha_{t-1})}{1-\bar{\alpha}_{t-1}}. && (41)
\end{aligned}
$$

A.4 Objectives of NoProp-DT and NoProp-CT

Starting from the ELBO derived in Appendix A.1, we have

$$
\begin{aligned}
\log p(y|x) &\geq \mathbb{E}_{q((z_t)_{t=0}^T|y,x)}\left[\log\frac{p((z_t)_{t=0}^T, y|x)}{q((z_t)_{t=0}^T|y,x)}\right] && (42)\\
&= \mathbb{E}_{q((z_t)_{t=0}^T|y)}\left[\log\frac{p(z_0)\left(\prod_{t=1}^T p(z_t|z_{t-1},x)\right)p(y|z_T)}{q(z_T|y)\left(\prod_{t=T}^1 q(z_{t-1}|z_t)\right)}\right] && (43)\\
&= \mathbb{E}_{q((z_t)_{t=0}^T|y)}\left[\log\frac{p(z_0)p(y|z_T)}{q(z_T|y)} + \log\prod_{t=1}^T\frac{p(z_t|z_{t-1},x)}{q(z_{t-1}|z_t,y)}\right] && (44)\\
&= \mathbb{E}_{q((z_t)_{t=0}^T|y)}\left[\log\frac{p(z_0)p(y|z_T)}{q(z_T|y)} + \log\prod_{t=1}^T\frac{p(z_t|z_{t-1},x)}{\frac{q(z_t|z_{t-1},y)q(z_{t-1}|y)}{q(z_t|y)}}\right] && (45)\\
&= \mathbb{E}_{q((z_t)_{t=0}^T|y)}\left[\log\frac{p(z_0)p(y|z_T)}{q(z_T|y)} + \log\frac{q(z_T|y)}{q(z_0|y)} + \log\prod_{t=1}^T\frac{p(z_t|z_{t-1},x)}{q(z_t|z_{t-1},y)}\right] && (46)\\
&= \mathbb{E}_{q((z_t)_{t=0}^T|y)}\left[\log\frac{p(z_0)p(y|z_T)}{q(z_0|y)} + \log\prod_{t=1}^T\frac{p(z_t|z_{t-1},x)}{q(z_t|z_{t-1},y)}\right] && (47)\\
&= \mathbb{E}_{q((z_t)_{t=0}^T|y)}\left[\log p(y|z_T)\right] + \mathbb{E}_{q((z_t)_{t=0}^T|y)}\left[\log\frac{p(z_0)}{q(z_0|y)}\right] + \sum_{t=1}^T\mathbb{E}_{q((z_t)_{t=0}^T|y)}\left[\log\frac{p(z_t|z_{t-1},x)}{q(z_t|z_{t-1},y)}\right] && (48)\\
&= \mathbb{E}_{q(z_T|y)}\left[\log p(y|z_T)\right] + \mathbb{E}_{q(z_0|y)}\left[\log\frac{p(z_0)}{q(z_0|y)}\right] + \sum_{t=1}^T\mathbb{E}_{q(z_{t-1},z_t|y)}\left[\log\frac{p(z_t|z_{t-1},x)}{q(z_t|z_{t-1},y)}\right] && (49)\\
&= \mathbb{E}_{q(z_T|y)}\left[\log p(y|z_T)\right] - D_{\mathrm{KL}}(q(z_0|y)\|p(z_0)) - \sum_{t=1}^T\mathbb{E}_{q(z_{t-1}|y)}\left[D_{\mathrm{KL}}(q(z_t|z_{t-1},y)\|p(z_t|z_{t-1},x))\right]. && (50)
\end{aligned}
$$

Since $q(z_t|z_{t-1}, y)$ and $p(z_t|z_{t-1}, x)$ are both Gaussian, with $q(z_t|z_{t-1}, y) = \mathcal{N}_d(z_t|\mu_t(z_{t-1}, u_y), c_t)$ and $p(z_t|z_{t-1}, x) = \mathcal{N}_d(z_t|\mu_t(z_{t-1}, \hat{u}_{\theta_t}(z_{t-1}, x)), c_t)$, their KL divergence is available in closed form:

$$D_{\mathrm{KL}}(q(z_t|z_{t-1}, y)\|p(z_t|z_{t-1}, x)) = \frac{1}{2c_t}\|\mu_t(z_{t-1}, \hat{u}_{\theta_t}(z_{t-1}, x)) - \mu_t(z_{t-1}, u_y)\|^2 \tag{51}$$

$$= \frac{1}{2c_t}\frac{\bar{\alpha}_t(1-\alpha_{t-1})^2}{(1-\bar{\alpha}_{t-1})^2}\|\hat{u}_{\theta_t}(z_{t-1}, x) - u_y\|^2 \tag{52}$$

$$= \frac{1-\bar{\alpha}_{t-1}}{2(1-\bar{\alpha}_t)(1-\alpha_{t-1})}\frac{\bar{\alpha}_t(1-\alpha_{t-1})^2}{(1-\bar{\alpha}_{t-1})^2}\|\hat{u}_{\theta_t}(z_{t-1}, x) - u_y\|^2 \tag{53}$$

$$= \frac{\bar{\alpha}_t(1-\alpha_{t-1})}{2(1-\bar{\alpha}_t)(1-\bar{\alpha}_{t-1})}\|\hat{u}_{\theta_t}(z_{t-1}, x) - u_y\|^2 \tag{54}$$

$$= \frac{1}{2}\left(\frac{\bar{\alpha}_t}{1-\bar{\alpha}_t} - \frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}}\right)\|\hat{u}_{\theta_t}(z_{t-1}, x) - u_y\|^2 \tag{55}$$

$$= \frac{1}{2}(\mathrm{SNR}(t) - \mathrm{SNR}(t-1))\|\hat{u}_{\theta_t}(z_{t-1}, x) - u_y\|^2. \tag{56}$$

Using $\hat{p}_{\theta_{\text{out}}}(y|z_T)$ to estimate $p(y|z_T)$, the ELBO becomes

$$\mathbb{E}_{q(z_T|y)}\left[\log \hat{p}_{\theta_{\text{out}}}(y|z_T)\right] - D_{\mathrm{KL}}(q(z_0|y)\|p(z_0)) - \frac{1}{2}\sum_{t=1}^{T}\mathbb{E}_{q(z_{t-1}|y)}\left[(\mathrm{SNR}(t) - \mathrm{SNR}(t-1))\|\hat{u}_{\theta_t}(z_{t-1}, x) - u_y\|^2\right]. \tag{57}$$

Instead of computing all T terms in the sum, we replace it with an unbiased estimator, which gives

$$\mathbb{E}_{q(z_T|y)}\left[\log \hat{p}_{\theta_{\text{out}}}(y|z_T)\right] - D_{\mathrm{KL}}(q(z_0|y)\|p(z_0)) - \frac{T}{2}\mathbb{E}_{t\sim\mathcal{U}\{1,T\}, q(z_{t-1}|y)}\left[(\mathrm{SNR}(t) - \mathrm{SNR}(t-1))\|\hat{u}_{\theta_t}(z_{t-1}, x) - u_y\|^2\right]. \tag{58}$$

With an additional hyperparameter η in Equation 8, this yields the NoProp objective in the discrete-time case. However, note that in our experiments, we choose to iterate over the values of t from 1 to T instead of sampling random values. Details can be found in Algorithm 1.

In the continuous-time case where t is scaled to the range (0, 1), let $\tau = 1/T$. If we rewrite the term involving SNR in the NoProp objective as

$$\frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\frac{\mathrm{SNR}(t+\tau) - \mathrm{SNR}(t)}{\tau}\|\hat{u}_\theta(z_t, x, t) - u_y\|^2\right]. \tag{59}$$

As $T \to \infty$, this becomes

$$\frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}\{0,1\}}\left[\mathrm{SNR}'(t)\|\hat{u}_\theta(z_t, x, t) - u_y\|^2\right]. \tag{60}$$

Again, with an additional hyperparameter η, this gives the NoProp objective in the continuous-time case in Equation 9.

B Trainable noise schedule for NoProp-CT

Following Kingma et al. (2021) and Gulrajani & Hashimoto (2024), we parameterize the signal-to-noise ratio (SNR) as

$$\mathrm{SNR}(t) = \exp(-\gamma(t)), \tag{65}$$

where $\gamma(t)$ is a learnable function that determines the rate of noise decay. To ensure consistency with our formulation, $\gamma(t)$ must be monotonically decreasing in t.

We implement $\gamma(t)$ using a neural network-based parameterization. Specifically, we define an intermediate function $\bar{\gamma}(t)$, which is normalized to the unit interval:

$$\bar{\gamma}(t) = \frac{\tilde{\gamma}(t) - \tilde{\gamma}(0)}{\tilde{\gamma}(1) - \tilde{\gamma}(0)}, \tag{66}$$

where $\tilde{\gamma}(t)$ is modeled as a two-layer neural network where the weights are restricted to be positive.

To align with our paper's formulation where $\gamma(t)$ should decrease with t, we define:

$$\gamma(t) = \gamma_0 + (\gamma_1 - \gamma_0)(1 - \bar{\gamma}(t)), \tag{67}$$

where γ_0 and γ_1 are trainable endpoints of the noise schedule. Finally, we obtain the noise schedule $\bar{\alpha}_t = \sigma(-\gamma(t))$.

Dataset	Method	Batch Size	Epochs	Optimiser	Learning Rate	Weight Decay	Timesteps	η
MNIST	Backprop	128	100	AdamW	0.001	0.001	10	-
	NoProp-DT	128	100	AdamW	0.001	0.001	10	0.1
	Adjoint	128	2	AdamW	0.001	0.001	1000	-
	NoProp-CT	128	100	Adam	0.001	0.001	1000	1
	NoProp-FM	128	100	Adam	0.001	0.001	1000	-
CIFAR-10	Backprop	128	150	AdamW	0.001	0.001	10	-
	NoProp-DT	128	150	AdamW	0.001	0.001	10	0.1
	Adjoint	128	4	AdamW	0.001	0.001	1000	-
	NoProp-CT	128	500	Adam	0.001	0.001	1000	1
	NoProp-FM	128	500	Adam	0.001	0.001	1000	-
CIFAR-100	Backprop	128	150	AdamW	0.001	0.001	10	-
	NoProp-DT	128	150	AdamW	0.001	0.001	10	0.1
	Adjoint	128	4	AdamW	0.001	0.001	1000	-
	NoProp-CT	128	1000	Adam	0.001	0.001	1000	1
	NoProp-FM	128	1000	Adam	0.001	0.001	1000	-

Table 3: Experiment details. η is the hyperparameter in Equations 8 and 9.

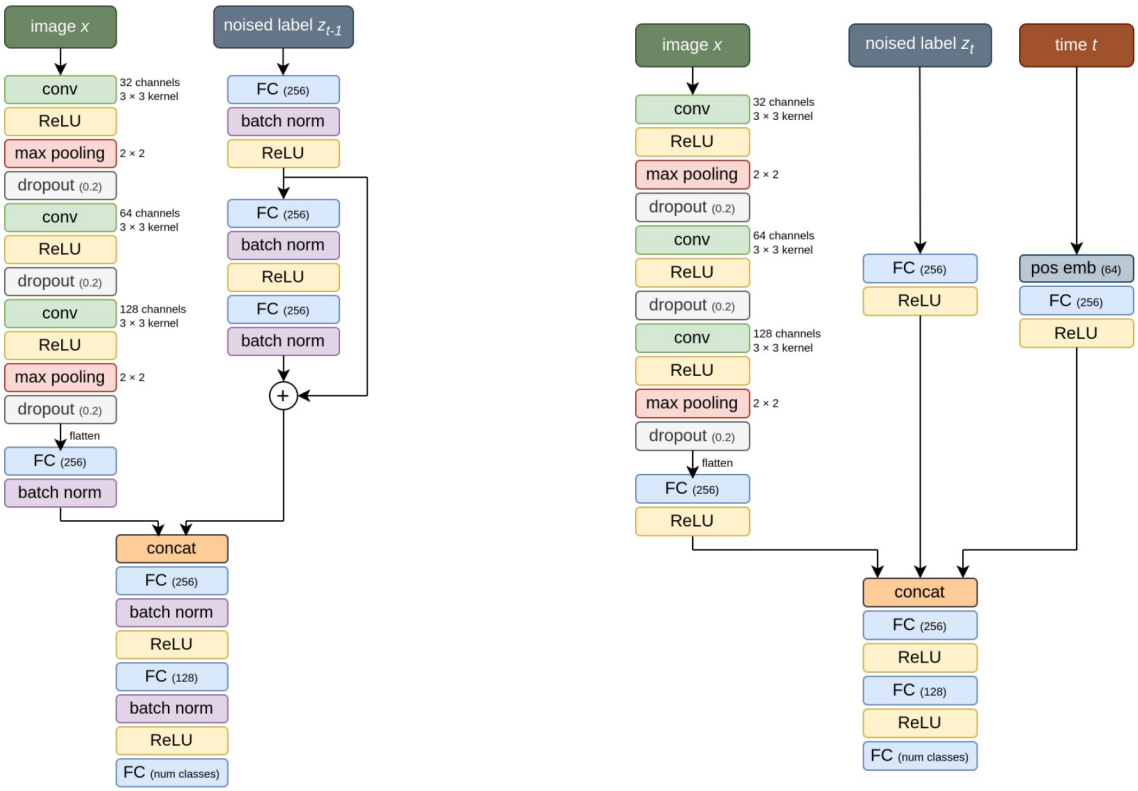

Figure 6: Models used for training when the class embedding dimension is different from the image dimension. Left: model for the discrete-time case. Right: model for the continuous-time case. conv: convolutional layer. FC: fully connected layer (number in parentheses indicates units). concat: concatenation. pos emb: positional embedding (number in parentheses indicates time embedding dimension). When the class embedding dimension matches the image dimension, the noised label and the image are processed in the same way before concatenation in each model. Note that batch normalization is not included in the continuous-time model.

Algorithm 1 NoProp-DT (Training)

Require: T diffusion steps, dataset $\{(x_i, y_i)\}_{i=1}^N$, batch size B, hyperparameter η, embedding matrix W_{Embed}, parameters $\{\theta_t\}_{t=1}^T, \theta_{\text{out}}$, noise schedule $\{\alpha_t\}_{t=0}^T$

for $t = 1$ to T **do**

  for each mini-batch $\mathcal{B} \subset \{(x_i, y_i)\}_{i=1}^N$ of size B **do**

    for each $(x_i, y_i) \in \mathcal{B}$ **do**

      Obtain label embedding $u_{y_i} = \{W_{\text{Embed}}\}_{y_i}$.

      Sample $z_{t,i} \sim \mathcal{N}_d(z_{t,i} | \sqrt{\bar{\alpha}_t} u_{y,i}, 1 - \bar{\alpha}_t)$.

    end for

    Compute the loss function:

$$\begin{aligned}\mathcal{L}_t = &\frac{1}{B} \sum_{i \in \mathcal{B}} [-\log \hat{p}_{\theta_{\text{out}}}(y_i | z_{T,i})] \\ &+ \frac{1}{B} \sum_{i \in \mathcal{B}} D_{\text{KL}}(q(z_0|y_i) \| p(z_0)) \\ &+ \frac{T}{2B} \eta \sum_{i \in \mathcal{B}} (\text{SNR}(t) - \text{SNR}(t-1)) \left\| \hat{u}_{\theta_t}(z_{t-1,i}, x_i) - u_{y_i} \right\|^2 . \end{aligned} \tag{61}$$

    Update θ_t, θ_{out}, and W_{Embed} using gradient-based optimization.

  end for

end for

Algorithm 2 NoProp-CT (Training)

Require: dataset $\{(x_i, y_i)\}_{i=1}^N$, batch size B, hyperparameter η, embedding matrix W_{Embed}, parameters $\theta, \theta_{\text{out}}$, noise schedule $\bar{\alpha}_t = \sigma(-\gamma_\psi(t))$

for each mini-batch $\mathcal{B} \subset \{(x_i, y_i)\}_{i=1}^N$ with size B **do**

  for each $(x_i, y_i) \in \mathcal{B}$ **do**

    Obtain label embedding $u_{y_i} = \{W_{\text{Embed}}\}_{y_i}$.

    Sample $t_i \sim \mathcal{U}(0, 1)$.

    Sample $z_{t_i,i} \sim \mathcal{N}_d(z_{t_i,i} | \sqrt{\bar{\alpha}_{t_i}} u_{y,i}, 1 - \bar{\alpha}_{t_i})$.

  end for

  Compute the loss function:

$$\begin{aligned}\mathcal{L} = &\frac{1}{B} \sum_{i \in \mathcal{B}} [-\log \hat{p}_{\theta_{\text{out}}}(y_i | z_{1,i})] \\ &+ \frac{1}{B} \sum_{i \in \mathcal{B}} D_{\text{KL}}(q(z_0|y_i) \| p(z_0)) \\ &+ \frac{1}{2B} \eta \sum_{i \in \mathcal{B}} \text{SNR}'(t_i) \left\| \hat{u}_\theta(z_{t_i,i}, x_i, t_i) - u_{y_i} \right\|^2 . \end{aligned} \tag{62}$$

  Update θ, θ_{out}, ψ, and W_{Embed} using gradient-based optimization.

end for

Algorithm 3 NoProp-FM (Training)

Require: dataset $\{(x_i, y_i)\}_{i=1}^N$, batch size B, embedding matrix W_{Embed}, parameters $\theta, \theta_{\text{out}}$

for each mini-batch $\mathcal{B} \subset \{(x_i, y_i)\}_{i=1}^N$ with size B **do**

for each $(x_i, y_i) \in \mathcal{B}$ **do**

Obtain label embedding $u_{y_i} = \{W_{\text{Embed}}\}_{y_i}$ and set $z_{1,i} = u_{y_i}$.

Sample $z_{0,i} \sim \mathcal{N}_d(z_{0,i}|0, 1)$.

Sample $t_i \sim \mathcal{U}(0, 1)$.

Sample $z_{t_i,i} \sim \mathcal{N}_d(z_{t_i,i} \mid t_i z_{1,i} + (1 - t_i) z_{0,i}, \sigma^2)$.

end for

Compute the loss function:

$$\mathcal{L} = \frac{1}{B} \sum_{i \in \mathcal{B}} \| v_\theta(z_{t_i,i}, x_i, t_i) - (z_{1,i} - z_{0,i}) \|^2. \tag{63}$$

if W_{Embed} has learnable parameters **then**

for each $(x_i, y_i) \in \mathcal{B}$ **do**

Compute the extrapolated linear estimate $\tilde{z}_{1,i} = z_{t_i,i} + (1 - t_i) v_\theta(z_{t_i,i}, x_i, t_i)$.

end for

Modify the loss function:

$$\mathcal{L} \leftarrow \mathcal{L} - \frac{1}{B} \sum_{i \in \mathcal{B}} \log \hat{p}_{\theta_{\text{out}}}(y_i \mid \tilde{z}_{1,i}). \tag{64}$$

end if

Update θ, θ_{out}, and W_{Embed} using gradient-based optimization.

end for